A New Automated Redistricting Simulator Using Markov Chain Monte Carlo

Benjamin Fifield†  Michael Higgins‡  Kosuke Imai§  Alexander Tarr¶

First draft: July 20, 2014
This draft: April 14, 2015

Abstract

Legislative redistricting is a critical element of representative democracy. A number of substantive scholars have used simulation methods to sample redistricting plans under various constraints in order to assess their impacts on partisanship and other aspects of representation. However, surprisingly few simulation methods exist in the literature, and the standard algorithm has no theoretical justification. To fill this gap, we propose a new automated redistricting simulator using Markov chain Monte Carlo. We formulate redistricting as a graph-cut problem and adopt the Swendsen-Wang algorithm for sampling contiguous districts. We then extend this basic algorithm to incorporate various constraints including equal population and geographical compactness. Finally, we apply simulated and parallel tempering to improve the mixing of the resulting Markov chain. The proposed algorithms, therefore, are designed to approximate the population of redistricting plans under various constraints. Through a small-scale validation study, we show that the proposed algorithm outperforms the existing standard algorithm. We also apply the proposed methodology to the data from New Hampshire and Mississippi. The open-source software is available for implementing the proposed methodology.

Keywords: gerrymandering, graph cuts, Metropolis-Hastings algorithm, simulated tempering, parallel tempering, Swendsen-Wang algorithm

* We thank Jowei Chen, Jacob Montgomery, and seminar participants at Dartmouth, Duke, and Microsoft Research (New York City Lab) for useful comments and suggestions. We thank James Lo, Jonathan Olmsted, and Radhika Saksena for their advice on computation. The open-source R package redist for implementing the proposed methodology is available as Fifield et al. (2015).
† Ph.D. Candidate, Department of Politics, Princeton University, Princeton NJ 08544. Email: bfifield@princeton.edu, URL: http://www.benfifield.com
‡ Postdoctoral Fellow, Department of Politics, Princeton University, Princeton NJ 08544. Email: mjh5@princeton.edu, URL: http://www.princeton.edu/~mjh5
§ Professor, Department of Politics, Princeton University, Princeton NJ 08544. Phone: 609–258–6601, Email: kimai@princeton.edu, URL: http://imai.princeton.edu
¶ Ph.D. Candidate, Department of Electrical Engineering, Princeton University, Princeton NJ 08544. Email: atarr@princeton.edu
1 Introduction

Legislative redistricting is a critical element of representative democracy. Previous studies have found that redistricting influences turnout and representation (e.g., Abramowitz 1983, Gelman and King 1994, Ansolabehere et al. 2000, McCarty et al. 2009, Barreto et al. 2004). From a public policy perspective, redistricting is potentially subject to partisan gerrymandering. After the controversial 2003 redistricting in Texas, for example, Republicans won 21 congressional seats in the 2004 election (Democrats won 11) whereas they had only 15 seats in 2002 (Democrats won 17). To address this concern, numerous remedies, including geographical compactness and partisan symmetry requirements, have been proposed (see Grofman and King 2007, Fryer and Holden 2011 and references therein).

The development of automated redistricting algorithms, which is the goal of this paper, began in the 1960s. Vickrey (1961) argued that such an “automatic and impersonal procedure” can eliminate gerrymandering (p. 110). After Baker v. Carr (1962) where the Supreme Court ruled that federal courts may review the constitutionality of state legislative apportionment, citizens, policy makers, and scholars became interested in redistricting. Weaver and Hess (1963) and Nagel (1965) were among the earliest attempts to develop automated redistricting algorithms (see also Hess et al. 1965). Since then, a large number of methods have been developed to find an optimal redistricting plan for a given set of criteria (e.g., Garfinkel and Nemhauser 1970; Browdy 1990; Bozkaya et al. 2003; Chou and Li 2006; Fryer and Holden 2011). These optimization methods serve as useful tools when drawing district boundaries (see Altman et al. 2005 for an overview).

However, the main interest of substantive scholars has been to characterize the distribution of possible redistricting plans under various criteria for detecting instances of
gerrymandering and understanding the causes and consequences of redistricting (e.g., Engstrom and Wildgen 1977; O’Loughlin 1982; Cirincione et al. 2000; McCarty et al. 2009; Chen and Rodden 2013). In 42 of the 50 U.S. states, for example, state politicians control the redistricting process and approve redistricting plans through standard statutory means. Therefore, an important institutional and policy policy question is how to effectively constrain these politicians through means such as compactness requirements (e.g., Niemi et al. 1990), in order to prevent the manipulation of redistricting for partisan ends. Simulation methods allow substantive scholars to answer these questions by approximating distributions of possible electoral outcomes under various institutional constraints.

Yet, surprisingly few simulation algorithms exist in the methodological literature. In fact, most, if not all, of these existing studies use essentially the same Monte Carlo simulation algorithm where a geographical unit is randomly selected as a “seed” for each district and then neighboring units are added to contiguously grow this district until it reaches the pre-specified population threshold (e.g., Cirincione et al. 2000; Chen and Rodden 2013). Unfortunately, no theoretical justification is given for these existing simulation algorithms, and some of them are best described as ad-hoc. A commonly used algorithm of this type is proposed by Cirincione et al. (2000) and implemented by Altman and McDonald (2011) in their open-source software. We hope to improve this state of the methodological literature.

To fulfill this methodological gap, in Section 2 we propose a new automated redistricting simulator using Markov chain Monte Carlo (MCMC). We formulate the task of drawing districting boundaries as the problem of graph-cuts, i.e., partitioning an adjacency graph into several connected subgraphs. We then adopt a version of the Swendsen-Wang algorithm to sample contiguous districts (Swendsen and Wang 1987; Barbu and Zhu 2005). We further extend this basic algorithm to incorporate
various constraints commonly imposed on redistricting plans, including equal population requirements and geographical compactness. Finally, we apply simulated and parallel tempering to improve the mixing of the resulting Markov chain \cite{Marinari1992, Geyer1995}. Therefore, unlike the existing algorithms, the proposed algorithms are designed to yield a representative sample of redistricting plans under various constraints. The open-source software, an R package \texttt{redist}, is available for implementing the proposed methodology \cite{Fifield2015}.

In Section 3, we conduct a small-scale validation study where all possible redistricting plans under various constraints can be enumerated in a reasonable amount of time. We show that the proposed algorithms successfully approximate this true population distribution while the standard algorithm fails even in this small-scale redistricting problem. We also conduct an empirical study in realistic settings using redistricting and U.S. Census data from New Hampshire and Mississippi. In this case, the computation of the true population distribution is not feasible and so we evaluate the empirical performance of the proposed algorithms by examining several standard diagnostics of MCMC algorithms. Lastly, Section 4 gives concluding remarks.

2 The Proposed Methodology

In this section, we describe the proposed methodology. We begin by formulating redistricting as a graph-cut problem. We then propose a Markov chain Monte Carlo (MCMC) algorithm to uniformly sample redistricting plans with $n$ contiguous districts. Next, we show how to incorporate various constraints such as equal population and geographical compactness. Finally, we improve the mixing of the MCMC algorithm by applying simulated and parallel tempering. A brief comparison with the existing algorithms is also given.
2.1 Redistricting as a Graph-cut Problem

Consider a typical redistricting problem where a state consisting of \( m \) geographical units (e.g., census blocks or voting precincts) must be divided into \( n \) contiguous districts. We formulate this redistricting problem as that of graph-cut where an adjacency graph is partitioned into a set of connected subgraphs \[ \text{[Altman, 1997]} \text{[Mehrotra et al., 1998]} \]. Formally, let \( G = \{V, E\} \) represent an adjacency graph where \( V = \{\{1\}, \{2\}, \ldots, \{m\}\} \) is the set of nodes (i.e., geographical units of redistricting) to be partitioned and \( E \) is the set of edges connecting neighboring nodes. This means that if two units, \( \{i\} \) and \( \{j\} \), are contiguous, there is an edge between their corresponding nodes on the graph, \( (i, j) \in E \).

Given this setup, redistricting can be seen equivalent to the problem of partitioning an adjacency graph \( G \). Formally, we partition the set of nodes \( V \) into \( n \) blocks, \( v = \{V_1, V_2, \ldots, V_n\} \) where each block is a non-empty subset of \( V \), and every node in \( V \) belongs to one and only one block, i.e., \( V_k \cap V_\ell = \emptyset \) and \( \bigcup_{k=1}^{n} V_k = V \). Such a partition \( v \) generates an adjacency subgraph \( G_v = (V, E_v) \) where \( E_v \) is a subset of \( E \). Specifically, an edge \( (i, j) \) belongs to \( E_v \) if and only if \( (i, j) \in E \) and nodes \( \{i\} \) and \( \{j\} \) are contained in the same block of the partition, i.e., \( \{i\}, \{j\} \in V_k \). Because \( E_v \) is obtained by removing some edges from \( E \) or “cutting” them, redistricting represents a graph cut problem. Finally, since each resulting district must be contiguous, a valid partition consists of only connected blocks where for any two nodes \( \{i\} \) and \( \{j\} \) in a connected block \( V_k \in v \), there exists a path of edges within \( V_k \) that joins these two nodes. Formally, there exists a set of nodes \( \{\{i\} = \{i_0\}, \{i_1\}, \{i_2\}, \ldots, \{i_{m'}-1\}, \{i_{m'}\} = \{j\}\} \subset V_k \) such that, for all \( \ell \in \{1, \ldots, m'\} \), \( (i_{\ell-1}, i_\ell) \in E_v \).

Figure [1] presents two illustrative examples, one of which is used in our validation study in Section [3.1]. These examples are taken from actual Florida precinct data in
Figure 1: Redistricting as a Graph-cut Problem. A state is represented by an adjacency graph where nodes are geographical units and edges between two nodes imply their contiguity. Under this setting, redistricting is equivalent to removing or cutting some edges (light grey) to form connected subgraphs, which correspond to districts. Different districts are represented by different colors. Two illustrative examples, one of which is used in our validation study in Section 3.1, are given here.

an attempt to create realistic, albeit small, examples. A state is represented by an adjacency graph where nodes are geographical units and edges between two nodes imply their contiguity. The figure demonstrates that redistricting a state into $n$ districts is equivalent to removing some edges of an adjacency graph (light grey) and forming $n$ connected subgraphs.

2.2 The Basic Algorithm for Sampling Contiguous Districts

We propose a new automated simulator to uniformly sample valid redistricting plans with $n$ contiguous districts. The contiguity of valid partitions dramatically increases the difficulty of developing such an algorithm. Intuitive methods for constructing partitions at random – e.g., randomly assigning precincts to districts – have a minuscule chance of yielding contiguous districts, and enumerating all partitions with contiguous districts is too large of a problem to be tractable in realistic redistricting settings. For more discussion, see Section 3.1.
Our MCMC algorithm is designed to obtain a dependent but representative sample from the uniform distribution of valid redistricting plans. In particular, we modify and extend Algorithm 1 of Barbu and Zhu (2005), which combines the Swendsen-Wang algorithm (Swendsen and Wang, 1987) with a Metropolis-Hastings step (Metropolis et al., 1953; Hastings, 1970). This algorithm begins with a valid partition \( v_0 \) (e.g., an actual redistricting plan adopted by the state) and transitions from a valid partition \( v_{t-1} \) to another partition \( v_t \) at each iteration \( t \). Here, we describe the basic algorithm for sampling contiguous districts. Later in the paper, we extend this basic algorithm in a couple of important ways so that common constraints imposed on redistricting can be incorporated and the algorithm can be applied to states with a larger number of districts.

Figure 2 illustrates our algorithm using the 50 precinct example with 3 districts given in the right panel of Figure 1. Our algorithm begins by randomly “turning on” edges in \( E_{v_{t-1}} \); each edge is turned on with probability \( q \). In the left upper plot of Figure 2, the edges that are turned on are indicated with darker grey. Next, we identify components that are connected through these “turned-on” edges and are on the boundaries of districts in \( v_{t-1} \). Each such connected component is indicated by a dotted polygon in the right upper plot. Third, among these, a subset of non-adjacent connected components are randomly selected as shown in the left lower plot (two in this case). These connected components are reassigned to adjacent districts to create a candidate partition. Finally, the acceptance probability is computed based on two kinds of edges from each of selected connected components, which are highlighted in the left lower plot: (1) “turned-on” edges, and (2) “turned-off” edges that are connected to adjacent districts. We accept or reject the candidate partition based on this probability.

Our algorithm guarantees that its stationary distribution is equal to the uniform
Figure 2: The Basic Algorithm for Sampling Contiguous Districts. The plots illustrate the proposed algorithm (Algorithm I) using the 50 precinct data given in the right panel of Figure 1. First, in the left upper plot, each edge other than those which are cut in Figure 1 is “turned on” (dark grey) independently with certain probability. Second, in the right upper plot, connected components on the boundaries are identified (dashed polygons). Third, in the left lower plot, a certain number of non-adjacent connected components on boundaries are randomly selected (dashed polygons) and the acceptance ratio is calculated by counting certain edges (colored edges). Finally, in the right lower plot, the proposed swap is accepted using the Metropolis-Hastings ratio.
distribution of all valid partitions, thereby yielding a uniformly sampled sequence of redistricting plans with contiguous districts. We now formally describe this algorithm.

**Algorithm 1 (Sampling Contiguous Redistricting Plans)** We initialize the algorithm by obtaining a valid partition \( v_0 = \{V_{10}, V_{20}, \ldots, V_{n0}\} \) and then repeat the following steps at each iteration \( t \),

**Step 1 (“Turn on” edges):** From the partition \( v_{t-1} = \{V_{1,t-1}, V_{2,t-1}, \ldots, V_{n,t-1}\} \), obtain the adjacency graph \( G_{v_{t-1}} = (V, E_{v_{t-1}}) \). Obtain the edge set \( E^*_{v_{t-1}} \subset E_{v_{t-1}} \) where each edge \( e \in E^*_{v_{t-1}} \) is independently added to \( E^*_{v_{t-1}} \) with probability \( q \).

**Step 2 (Gather connected components on boundaries):** Find all components that are connected within \( E^*_{v_{t-1}} \) and adjacent to another block in the partition \( v_{t-1} \). Let \( C \) denote this set of connected components where for all \( C_\ell \in C \), there exists \( k \in \{1, 2, \ldots, n\} \) such that \( C_\ell \cap V_{k,t-1} = \emptyset \) and \((i, j) \in E\) for some \( \{i\} \in C_\ell \) and \( \{j\} \in V_{k,t-1} \).

**Step 3 (Select non-adjacent connected components):** Randomly select a set of \( r \) non-adjacent connected components \( C^* \) from \( C \) such that \( v_{t-1} \setminus C^* \) is a valid partition where each block of nodes \( V_{k,t-1} \setminus C^* \) is connected in \( G_{v_{t-1}} \). The sampling is done such that each eligible subset of \( C \) is selected with equal probability.

**Step 4 (Propose swaps):** Initialize a candidate partition \( v^*_t = (V^*_{1t}, V^*_{2t}, \ldots, V^*_{nt}) \) by setting \( V^*_{kt} = V_{k,t-1} \). For each component \( C^*_\ell \in C^* \) with \( \ell \in \{1, \ldots, r\} \), find the block \( V^*_{k,t} \in v_{t-1} \) that contains \( C^*_\ell \), and let \( A(C^*_\ell, v_{t-1}) \) denote the set of blocks in \( v_{t-1} \) that are adjacent to \( C^*_\ell \), not including the block that contains \( C^*_\ell \). Propose to assign \( C^*_\ell \) from block \( V^*_{k,t} \) to an adjacent block \( V^*_{j',t} \) with probability \( 1/|A(C^*_\ell, v_{t-1})| \). If \( C^*_\ell \) is assigned to block \( V^*_{k',t} \), set \( V^*_{kt} = V^*_{k',t-1} \cup C^*_\ell \) and \( V^*_{kt} = V^*_{k,t-1} \setminus C^*_\ell \). If \( V^*_{kt} = \emptyset \), go back to Step 3. Observe that, after each proposed swap, \( v^*_t \) remains a connected set partition.

**Step 5 (Accept or reject the proposal):** Set

\[
 v_t = \begin{cases} 
 v^*_t, & \text{with probability } \alpha(v_{t-1} \rightarrow v^*_t), \\
 v_{t-1}, & \text{with probability } 1 - \alpha(v_{t-1} \rightarrow v^*_t). 
\end{cases} 
\]  

The acceptance probability is given by the Metropolis criterion

\[
\alpha(v_{t-1} \rightarrow v^*_t) = \min \left( 1, (1 - q)^{|B(C^*, E_{v^*_t})| - |B(C^*, E_{v_{t-1}})|} \right) 
\]
where \( B(C^*, E_v) = \{(i, j) \in E_v : \exists C_t^* \in C^*, C_t^* \subset V_k \in v \text{ s.t. } \{i\} \in C_t^*, \{j\} \in V_k \setminus C_t^* \} \) denotes the set of edges in \( E_v \) that need to be cut to form connected components \( C^* \).

In the Appendix, we prove the following theorem, which states that if the Markov chain produced by the proposed algorithm is ergodic, the stationary distribution of the chain is uniform on the population of all valid partitions \( \Omega(G, n) \) (Tierney [1994]).

**Theorem 1** If every valid partition can be obtained through a sequence of moves given by Algorithm 1, then the stationary distribution of the resulting Markov chain is uniform on all valid partitions.

The acceptance ratio given in equation (2) is based on the Metropolis-Hastings detailed balance condition (Metropolis et al., 1953; Hastings, 1970),

\[
\alpha(v_{t-1} \rightarrow v_t^*) = \min \left( 1, \frac{\pi(v_t^* \rightarrow v_{t-1})}{\pi(v_{t-1} \rightarrow v_t^*)} \right),
\]

where \( \pi(v \rightarrow v^*) \) denote the probability that, starting from partition \( v \), an iteration of Algorithm 1 described above obtains a candidate partition \( v^* \) through Steps 1–4. Computing numerators and denominators of this ratio separately is combinatorially expensive. However, following Barbu and Zhu (2005), we show in the Appendix that substantial cancellation occurs, yielding a simple expression given in equation (2). Indeed, we only need to find all edges within \( E_{v_{t-1}} \) and \( E_{v_t^*} \) that join a node in a connected component of \( C_t^* \in C^* \) to a node not contained in the block. Since components in \( C^* \) are not adjacent, this will ensure that the node not contained in \( C_t^* \) will not be contained in a block in \( C^* \).

Several additional remarks are in order. First, when implementing this algorithm, Step 2 requires the three operations: (1) identify all nodes that form a boundary of multiple partitions by comparing \( G_{v_{t-1}} \) with the original adjacency graph \( G \), (2) identify all connected components that include at least one such node via the breadth-
first or depth-first search algorithm, and (3) identify the partition to which each
connected component belongs.

Second, in Step 3, we typically choose a positive integer $r$ by randomly sampling
it from a distribution with $\Pr(r = 1) > 0$ at each iteration. If $r = 1$, then the
ergodicity of the Markov chain is guaranteed but the algorithm moves slowly in the
sample space. When $r > 1$, the algorithm can mix faster by proposing multiple swaps.
However, depending on the adjacency graph $G$, the algorithm may fail to reach some
valid partitions. Thus, we allow $r$ to take a value greater than 1 while keeping the
probability of $r = 1$ positive (e.g., a truncated poisson distribution).

Third, in the original algorithm of Barbu and Zhu (2005), $r$ is set to 1 and instead
the authors use a small value of $q$ to create larger connected components. This
alternative strategy to improving mixing of the algorithm, though sensible in other
settings, is not applicable to the current case. The reason is that larger connected
components typically include more units from the interior of each block. This in turn
dramatically lowers the acceptance probability.

Finally, while this basic algorithm yields a sample of redistricting plans with con-
tiguous districts, it does not incorporate common constraints imposed on redistricting
process, including equal population and geographical compactness. In addition, our
experience shows that the algorithm does not scale for states with a medium or larger
number of districts. Therefore, we now describe two important modifications to the
basic algorithm.

### 2.3 Constraints and Reweighting

In a typical redistricting process, several additional constraints are imposed. Two
most commonly applied constraints are equal population and geographical compact-
ness. We first consider the equal population constraint. Suppose that we use $p_i$ to
denote the population size for node $\{i\}$ where the population parity for the state is
given by \( \bar{p} \equiv \frac{\sum_{i=1}^{m} p_i}{n} \). Then, the population equality constraint can be written as,

\[
P_v = \max_{1 \leq k \leq n} \left| \frac{\sum_{i \in V_k} p_i}{\bar{p}} - 1 \right| \leq \delta
\]  

(4)

where \( \delta \) determines the degree to which one wishes to impose the constraint. For example, \( \delta = 0.03 \) implies that the population of all districts must be within 3% of the population parity.

Next, we consider the geographical compactness. No consensus exists about the exact meaning of compactness and several alternative definitions have been proposed in the literature (see Niemi et al., 1990). Here, we adopt the measure recently proposed by Fryer and Holden (2011). Let \( w_i \) be the population density of node \( \{i\} \) and \( d_{ij} \) represent the distance between the centroids of nodes \( \{i\} \) and \( \{j\} \). The measure, which is called the relative proximity index, is based on the sum of squared distances among voters in each district relative to its minimum value. Then, the compactness constraint can be written as,

\[
R_v = \sum_{k=1}^{n} \sum_{i,j \in V_k, i < j} w_i w_j d_{ij}^2 \geq \min_{v' \in \Omega(G,n)} \sum_{k=1}^{n} \sum_{i,j \in V'_k, i < j} w_i w_j d_{ij}^2 \leq \epsilon
\]  

(5)

where \( V'_k \in v', \epsilon \) determines the strength of this constraint, and \( \Omega(G,n) \) is the set of all redistricting plans with \( n \) contiguous districts. Fryer and Holden (2011) develops an approximate algorithm to efficiently compute the minimum of the sum of squared distances, i.e., the denominator of equation (5). The authors also show that this measure is invariant to geographical size, population density, and the number of districts of a state, thereby allowing researchers to compare the index across different states and time periods.

How can we uniformly sample redistricting plans under these additional constraints? One possibility is to discard any candidate partition that does not satisfy the desired constraints. In Algorithm 1, after Step 4, one could check whether the candidate partition \( v^*_t \) satisfies the constraints and if not go back to Step 3. However,
such a strategy often dramatically slows down the algorithm and worsens mixing. Alternatively, researchers could run Algorithm 1 without any modification and then simply discard any sampled redistricting plans that do not meet the constraints. The problem of this approach is that many sampled plans may be discarded when strong constraints are imposed.

To overcome this difficulty, we propose to modify Algorithm 1 in the following manner. We first oversample the redistricting plans that are likely to meet the constraints. This means that fewer sampled plans are discarded due to the failure to satisfy the constraints. We then reweight the remaining valid redistricting plans such that they together approximate the uniform sampling from the population of all valid redistricting plans under the constraints. To do this, we consider the Gibbs distribution from statistical physics,

$$ P(v) = \frac{1}{z(\beta)} \exp \left( -\beta \sum_{V_k \in v} \psi(V_k) \right) $$

(6)

where $\beta \geq 0$ is the inverse temperature and $z(\beta)$ is the normalizing constant. The function $\psi(\cdot)$ is chosen so that it reflects the constraint of interest. For example, we use $\psi(V_k) = | \sum_{i \in V_k} p_i / \bar{p} - 1 |$ and $\psi(V_k) = \sum_{i,j \in V_k} w_i w_j d_{ij}^2$ for the equal population and geographical compactness constraints, respectively.

Algorithm 1 can be modified easily to sample from the non-uniform stationary distribution given in equation (6). In particular, we only need to change the acceptance probability in equation (2) of Step 5 to,

$$ \alpha(v_{t-1} \rightarrow v_t^*) = \min \left( 1, \frac{g_{\beta}(v_t^*)}{g_{\beta}(v_{t-1})} \cdot (1 - q)^{|B(C^*, v_t^*)| - |B(C^*, v_{t-1})|} \right) $$

(7)

where $g_{\beta}(v) \equiv \exp \left( -\beta \sum_{V_k \in v} \psi(V_k) \right)$. Lastly, we reweight the sampled plans by $1/g_{\beta}(v)$ to approximate the uniform sampling from the population of all possible valid redistricting plans. If we resample the sampled plans with replacement using this importance weight, then the procedure is equivalent to the sampling/importance
resampling (SIR) algorithm (Rubin 1987).

2.4 Simulated and Parallel Tempering

One major drawback of the reweighting approach is that when each plan is weighted according to equation (6) the algorithm may have a harder time moving through the sample space. We use simulated and parallel tempering to improve the mixing of Algorithm 1 in such situations (Marinari and Parisi 1992; Geyer and Thompson 1995). We begin by describing how to apply simulated tempering in this context.

Recall that we want to draw from the distribution given in equation (6). We initialize a sequence of inverse temperatures $\{\beta^{(\ell)}\}_{\ell=0}^{r-1}$ where $\beta^{(0)}$ corresponds to the cold temperature, which is the target parameter value for inference, and $\beta^{(r-1)} = 0$ represents the hot temperature with $\beta^{(0)} > \beta^{(1)} > \cdots > \beta^{(r-1)} = 0$. After many iterations, we keep the MCMC draws obtained when $\beta = \beta^{(0)}$ and discard the rest. By sampling under warm temperatures, simulated tempering allows for greater exploration of the target distribution. We then reweight the draws by the importance weight $1/g_{\beta^{(0)}(v)}$.

Specifically, we perform simulated tempering in two steps. First, we run an iteration of Algorithm 1 using the modified acceptance probability with $\beta = \beta^{(\ell)}$. We then make another Metropolis-Hastings decision on whether to change to a different value of $\beta$. The details of the algorithm are given below.

**Algorithm 2 (Simulated Tempering)** Given the initial valid partition $v_0$ and the initial temperature value $\beta_0 = \beta^{(k_0)}$ with $k_0 = r - 1$, the simulated tempering algorithm repeats the following steps at each iteration $t$.

**Step 1 (Run the basic algorithm with the modified acceptance probability):** Using the current partition $v_{t-1}$ and the current temperature $\beta_{t-1} = \beta^{(k_{t-1})}$, obtain a valid partition $v_t$ by running one iteration of Algorithm 1 with the acceptance probability given in equation (7).

**Step 2 (Choose a candidate temperature):** We set $k_t^* = k_{t-1} - 1$ with probability $u(k_{t-1}, k_{t-1} - 1)$ and set $k_t^* = k_{t-1} + 1$ with probability $u(k_{t-1}, k_{t-1} + 1)$.
1) = 1 − u(κ_{t-1}, κ_{t-1} - 1) where u(κ_{t-1}, κ_{t-1} - 1) = u(κ_{t-1}, κ_{t-1} + 1) = 1/2 when 1 ≤ κ_{t-1} ≤ r - 2, and u(r - 1, r - 2) = u(0, 1) = 1, u(r - 1, r) = u(0, -1) = 0.

**Step 3 (Accept or reject the candidate temperature):** Set

\[
κ_t = \begin{cases} 
κ^*_t, & \text{with probability } \gamma(κ_{t-1} \rightarrow κ^*_t), \\
κ_{t-1}, & \text{with probability } 1 - \gamma(κ_{t-1} \rightarrow κ^*_t) 
\end{cases}
\] (8)

where

\[
\gamma(κ_{t-1} \rightarrow κ^*_t) = \min \left( 1, \frac{g_β(κ^*_t)(v_t)}{g_β(κ_{t-1})(v_t)} \frac{u(κ^*_t, κ_{t-1}) w_{κ^*_t}}{u(κ_{t-1}, κ^*_t) w_{κ_{t-1}}} \right) \] (9)

where \( w_ℓ \) is an optional weight given to each \( l \in \{0, 1, \ldots, r-1\} \).

Much like simulated tempering, parallel tempering is also useful for improving mixing in MCMC algorithms and for sampling from multimodal distributions (Geyer, 1991). Parallel tempering differs from simulated tempering in that instead of varying the temperature within a single Markov chain, we run \( r \) copies of Algorithm 1 at \( r \) different temperatures, and after a fixed number of iterations we exchange the corresponding temperatures between two randomly selected adjacent chains using the Metropolis criterion. This algorithm has an advantage over Algorithm 2 in that we do not need to choose the prior probability of \( β \), which typically has a significant effect on the mixing performance. However this advantage comes at the expense of increased computation as we are now running \( r \) chains instead of just one.

The nature of parallel tempering suggests that it should be implemented in a parallel architecture, which can be used to minimize computation time. Altekar et al. (2004) describe such an implementation using parallel computing and MPI, which we use as the basis for implementing our algorithm described below.

**Algorithm 3 (Parallel Tempering)** Given \( r \) initial valid partitions \( v_0^{0}, v_0^{1}, \ldots, v_0^{r-1} \) and a sequence of \( r \) decreasing temperatures \( β^{(0)} > β^{(1)} > \cdots > β^{(r-1)} = 0 \) with \( β^{(0)} \) the target temperature for inference, and swapping interval \( T \), the parallel tempering algorithm repeats the following steps every iteration \( t \in \{0, T, 2T, 3T, \ldots\} \),

**Step 1 (Run the basic algorithm with the modified acceptance probability):** For each chain \( i \in \{0, 1, \ldots, r-1\} \), using the current partition \( v_t^{(i)} \) and
the corresponding temperature $\beta^{(i)}$, obtain a valid partition $v_{i+T}^{(i)}$ by running $T$ iterations of Algorithm [2] with the acceptance probability given in equation [7]. This step is executed concurrently for each chain.

**Step 2 (Propose a temperature exchange between two chains):** Randomly select two adjacent chains $j$ and $k$ and exchange information about the temperatures $\beta^{(j)}$, $\beta^{(k)}$ and the unnormalized likelihoods of the current partitions $g_{\beta^{(j)}}(v_{i+T}^{(j)})$, $g_{\beta^{(k)}}(v_{i+T}^{(k)})$ using MPI.

**Step 3 (Accept or reject the temperature exchange):** Exchange temperatures (i.e., $\beta^{(j)} \leftrightarrow \beta^{(k)}$) with probability

$$
\gamma(\beta^{(j)} \leftrightarrow \beta^{(k)}) = \min\left(1, \frac{g_{\beta^{(j)}}(v_{i+T}^{(k)}) g_{\beta^{(k)}}(v_{i+T}^{(j)})}{g_{\beta^{(j)}}(v_{i+T}^{(j)}) g_{\beta^{(k)}}(v_{i+T}^{(k)})}\right)
$$

(10)

All previously generated samples are assumed to have been generated at the current temperature of the chain.

We note that the mixing performance of Algorithm [3] is affected by the choice of the temperature sequence $\beta^{(i)}$. While no sequence has been shown to be optimal in the literature, sequences with power-law spacing have been shown heuristically to produce reasonable results. For this reason, we used the sequence $\beta^{(i)} = (\beta^{(0)})^{r-i}$, $i \in \{0, 1, \ldots, r - 1\}$ for our implementation.

### 2.5 Comparison with the Existing Algorithms

A number of substantive researchers used Monte Carlo simulation algorithms to sample possible redistricting plans under various criteria in order to detect the instances of gerrymandering and understand the causes and consequences of redistricting (e.g., Engstrom and Wildgen [1977], O’Loughlin [1982], Cirincione et al. [2000], McCarty et al. [2009], Chen and Rodden [2013]). Most of these studies use a similar Monte Carlo simulation algorithm where a geographical unit is randomly selected as a “seed” for each district and then neighboring units are added to contiguously grow this district until it reaches the pre-specified population threshold. A representative of such algorithms, proposed by Cirincione et al. (2000) and implemented by Altman and McDonald [2011] in their open-source BARD package, is given here.
Algorithm 4 (The Standard Redistricting Simulator (Cirincione et al., 2000))

For each district, we repeat the following steps.

**Step 1:** From the set of unassigned units, randomly select the seed unit of the district.

**Step 2:** Identify all unassigned units adjacent to the district.

**Step 3:** Randomly select one of the adjacent units and add it to the district.

**Step 4:** Repeat Steps 2 and 3 until the district reaches the predetermined population threshold.

Additional criteria can be incorporated into this algorithm by modifying Step 3 to select certain units. For example, to improve the compactness of the resulting districts, one may choose an adjacent unassigned unit that falls entirely within the minimum bounding rectangle of the emerging district. Alternatively, an adjacent unassigned unit that is the closest to emerging district can be selected (see Chen and Rodden, 2013).

Nevertheless, the major problem of these simulation algorithms is their adhoc nature. For example, as the documentation of BARD package warns, the creation of earlier districts may make it impossible to yield contiguous districts. More importantly, the algorithms come with no theoretical result and are not even designed to uniformly sample redistricting plans even though researchers have a tendency to assume that they are. In contrast, the proposed algorithms described in Sections 2.2 and 2.4 are built upon the well-known theories and strategies developed in the literature on the Markov chain Monte Carlo methods. The disadvantage of our algorithms, however, is that they yield a dependent sample and hence their performance will hinge upon the degree of mixing. Thus, we now turn to the assessment of the empirical performance of the proposed algorithms.
3 Empirical Performance of the Proposed Algorithms

In this section, we assess the performance of the proposed algorithms in two ways. First, we conduct a small-scale validation study where, due to its size, all possible redistricting maps can be enumerated in a reasonable amount of time. We show that our algorithms can approximate the target distribution well when the standard algorithm commonly used in the literature fails. Second, we use the actual redistricting data to examine the convergence behavior of the proposed algorithms in more realistic settings using the redistricting data from New Hampshire (two districts) and Mississippi (four districts). For these data, the computation of the true population distribution is not feasible. Instead, we evaluate the empirical performance of the proposed algorithms by examining the standard diagnostics of MCMC algorithms.

To conduct these analyses, we integrate precinct-level data from two sources. We utilize precinct-level shape files and electoral returns data from the Harvard Election Data Archive to determine precinct adjacency and voting behavior. We supplement this data with basic demographic information from the U.S. Census Bureau P.L. 94–171 summary files, which are compiled by the Census Bureau and disseminated to the 50 states in order to obtain population parity in decennial redistricting.

3.1 A Small-scale Validation Study

We conduct a validation study where we analyze the convergence of our algorithm to the target distribution on the 25 precinct set, which is shown as an adjacency graph in Figure 1. Due to the small size of these sets, all possible redistricting plans can be enumerated in a reasonable amount of time. We begin by considering the problem of partitioning each of these graphs into two districts. We apply the proposed algorithm
(Algorithm 1) with the starting map obtained randomly by running the standard algorithm (Algorithm 4) once. In addition, we apply the standard algorithm, as implemented in the BARD package [Altman and McDonald, 2011], to compare its performance with that of our proposed algorithm. We then consider partitions of the 25 precinct set into three districts. The results of the proposed algorithm are based on a single chain of 10,000 draws while those of the standard algorithm are based on the same number of independent draws.

Before we give results, it should be noted that, even for this small-scale study, the enumeration of all valid partitions is a non-trivial problem. For partitions of 25 precincts into three districts, of the roughly $3^{25}/6 \approx 1.41 \times 10^{11}$ possible partitions, 82,623 have three contiguous districts, and 3,617 have district populations within 20% of parity.

A brief description of our enumeration algorithm is as follows. In the case of two districts, we choose an initial starting node and form a partition where one district is that initial node and the other district is the complement, provided the complement is connected. We then form connected components of two nodes comprised of that starting node and and nodes that are adjacent to that node. We identify all valid partitions where one district is a two-node component and the other district is the complement of the component. We continue forming connected components of incrementally increasing sizes and finding valid partitions until all possible partitions are found. In the case of three precincts, if the complement of a connected component is comprised of two additional connected components, we store that partition as valid. If the complement is a single connected component, we apply the two-district algorithm on the complement. After this enumeration, we identify which partitions have districts with populations within a certain percentage of parity.

Figure 3 presents the results of the validation study with three districts and 25
Figure 3: A Small-scale Validation Study with Three Districts. The underlying data is the 25 precinct set shown in the left plot of Figure 1. The plots in the first row show that the proposed algorithm (Algorithm 1; solid black lines) approximates well the true population distribution (grey histograms) when no (left plot) or weak (middle plot) equal population constraint is imposed. However, the algorithm exhibits poor performance when a stronger constraint (right plot) is imposed. In contrast, the standard algorithm (Algorithm 4; red dashed lines) fails to approximate the target distribution in all cases. Finally, the plots in the second row show that the proposed algorithm with simulated tempering (Algorithm 2; black dot-dashed line) approximates the true population distribution well even when a stronger constraint is placed. The same exact pattern is observed for the parallel tempering algorithm (Algorithm 3; blue solid line). The results for each algorithm is based on a single chain of 10,000 draws.
precincts. We apply the proposed algorithm (Algorithm 1) with the starting map obtained randomly from the standard algorithm (Algorithm 4) (upper panel). These algorithms are also implemented with the simulated tempering (Algorithm 2; black dot-dashed lines) and parallel tempering (Algorithm 3; blue solid lines) strategies (the lower panel).

To implement these algorithms, we specify a sequence of temperatures \( \{\beta(\ell)\}_{\ell=0}^r \). For the population deviation of 20%, we chose a target temperature of \( \beta(r) = 5.4 \), and for the population deviation of 10%, we chose a target temperature of \( \beta(r) = 9 \). In both cases, we use \( \beta(0) = 0 \). We choose these setups so that the rejection ratio is in the recommended 20–40\% range (Geyer and Thompson, 1995) and the target temperature value is chosen based on the number of plans that meet the population constraint. In both cases, we use a subset of draws taken under the target temperature. We then resample the remaining draws using the importance weights \( 1/g_{\beta(\ell)}(v) \), and finally subset down to the set of remaining draws that fall within the population target.

The left-upper plot of Figure 3 shows that when no constraint is imposed the proposed algorithm approximates the target distribution well while the sample from the standard algorithm is far from being representative of the population. In the plots of the middle and right columns, we impose the equal population constraint where only up to 20\% and 10\% deviation from the population parity is allowed, respectively. It is no surprise that the standard algorithm completely fails to approximate the true distribution as well in these cases (the middle and right plots in the upper panel). In contrast, the proposed algorithms with simulated and parallel tempering approximate the true population distribution well. Even when a stronger constraint, i.e., 10\%, is placed, the proposed algorithms with simulated tempering (Algorithm 2) and parallel tempering (Algorithm 3) maintain a good approximation.

Finally, Figure 4 compares the runtime between the proposed basic algorithm
(Algorithm I solid black lines) and the standard algorithm (Algorithm IV red dashed lines) under various validation study settings. In addition to the 25 precinct set, we also include the 50 precinct set, which is shown in the right plot of Figure 4. Each algorithm is run until it yields 10,000 draws using a node on a Linux server with 2.66 GHz Nehalem processors and 3GB RAM (no parallel computing is used). We find that under all settings we consider here the runtime for the proposed algorithm is at least 50 times shorter than that for the standard algorithm. This difference increases as the number of precincts and the strength of equal population constraint (x-axis) increase. In sum, in terms of computational speed, the proposed algorithm scales much better than the standard algorithm.

3.2 An Empirical Study

The scale of the validation study presented above is small so that we can enumerate all possible redistricting plans in a reasonable amount of time. This allowed us to examine how well each algorithm is able to approximate the true population distri-
Figure 5: Precinct-level Maps of New Hampshire (327 precincts, two congressional districts) and Mississippi (1,969 precincts, four congressional districts). Colors correspond to precinct congressional district assignments in 2010. In New Hampshire, Democrats and Republicans each hold a single congressional seat. In Mississippi, Republicans hold three congressional seats while Democrats hold a single seat.

However, the scale of the study is too small to be realistic. Below, we apply the proposed algorithms to the 2008 election data and conduct standard convergence diagnostics of MCMC algorithms. While we cannot compare the distribution of sampled maps with the true population distribution, this empirical study enables us to investigate the performance of the proposed methods in realistic settings.

New Hampshire. We first consider New Hampshire. The state has two congressional districts and consists of 327 precincts, and so this is one of the simplest realistic redistricting problems. The left panel of Figure 5 shows the implemented statewide redistricting plan as of 2010. Under this plan, Democrats and Republicans won a single congressional seat each. In 2008, Obama won 54% of votes in this state while his 2012 vote share was 52%. Redistricting in New Hampshire is determined by its state legislature and plans are passed as standard statutes, which makes them subject to gubernatorial veto. We apply the proposed basic algorithm (Algorithm 1), simulated
tempering algorithm (Algorithm 2), and parallel tempering algorithm (Algorithm 3). The target population consists of all redistricting plans with contiguous districts and a maximum of 1% deviation from the population parity.

A total of 10 chains are run until 500,000 draws are obtained for each of the three algorithms. Inference is based on a total of 22,970 draws, which is the lowest number of draws across the three algorithms that both satisfy the population constraint and were drawn under the target temperature value, $\beta^{(r)} = 27$. For starting values, we use independent draws from the standard algorithm (Algorithm 4 as implemented in the BARD package). For both the simulated and parallel tempering algorithms, after some preliminary analysis, we have decided to allow $\beta^{(t)}$ to take values between 0 and 27, using power-law spacing, with the target temperature value of 27. As in the small-scale verification study, we only use draws taken under the target temperature, and then reweight according to the importance weights $1/g_{\beta^{(t)}(v)}$ before selecting all remaining draws that fall within the target parity deviation of 1%.

Figure 6 presents the results. The figure shows the autocorrelation plots (left column), the trace plots (middle column), and the Gelman-Rubin potential scale reduction factors (Gelman and Rubin, 1992; right column) for the basic algorithm (top panel), the simulated tempering algorithm (middle panel) and the parallel tempering algorithm (bottom panel). We use the logit transformed Republican dissimilarity index for all diagnostics. Both the simulated and parallel tempering algorithms significantly outperform the basic algorithm. The former has a lower autocorrelation and mixes better. In addition, the potential scale reduction factor goes down quickly, suggesting that all the chains with different starting maps become indistinguishable from each other after approximately 1,500 draws.

Mississippi. Next, we analyze the 2008 election data from Mississippi. This state has a total of four congressional districts and 1,969 precincts, thereby providing a
Figure 6: Convergence Diagnostics of the Proposed Algorithm for the 2008 New Hampshire Redistricting Data. The proposed basic algorithm (Algorithm 1, top panel), the simulated tempering algorithm (Algorithm 2, middle panel), and the parallel tempering algorithm (Algorithm 3, bottom panel) are applied to the New Hampshire data with 327 precincts and 2 congressional districts. The target population consists of all redistricting plans with contiguous districts and a maximum of 1% deviation from the population parity. A total of 10 chains are run with different starting maps for each algorithm until 500,000 draws are obtained, and inference is based on a total of 22,970 draws (the number of draws in the simulated tempering algorithm that are both drawn under the target temperature and satisfy the target population constraint). For the logit transformed Republican dissimilarity index, the autocorrelation plots (left column), the trace plots (middle column), and the Gelman-Rubin potential scale reduction factors (right column) are presented. The simulated and parallel tempering algorithms outperform the basic algorithm across all three diagnostics.
more challenging example when compared to New Hampshire. The right-hand panel of Figure 5 shows the implemented redistricting plan in Mississippi as of 2010. In 2008, 43% of the electorate voted for Obama while his votes share in the 2012 election for this state was 44%. Redistricting in Mississippi is determined by its state legislature subject to gubernatorial veto.

One important feature of Mississippi is its sizable African-American population (37% of the population). This group is concentrated in the capital city, Jackson, and in surrounding areas in the west of the state, which poses a special challenge to the algorithms. Democrats typically win this seat, shaded in blue in Figure 5, while Republicans typically win the other three seats in Mississippi. Mississippi is also one of the nine states fully covered by Section V of the Voting Rights Act, which obligates political officials to submit its proposed redistricting plan to the U.S. Department of Justice. However, following the Supreme Court’s decision in Shelby County v. Holder (2013) to strike down the pre-clearance formula determining Section V coverage, Mississippi is no longer subject to Section V requirements by default.

Here, we utilize parallel tempering (Algorithm 3) to examine its algorithmic performance for Mississippi. After some preliminary analysis, we chose to anneal $\beta^{(\ell)}$ between 0 and $-225$ in unequally spaced increments, with the target temperature of $\beta^{(\ell)} = -225$. We run a total of 10 chains for 200,000 simulations each, keeping every 5th draw. Inference is then based off of a total of 138,840 draws, which is the number of remaining simulations drawn under the target $\beta^{(\ell)}$ that fall within 5% of population parity.

Figure 7 presents the results of this analysis. The same set of diagnostics are conducted for the Republican dissimilarity index (top row) and the African-American dissimilarity index (bottom row). The figure shows that although the Mississippi data pose a much more challenging application than the New Hampshire data, the
parallel tempering algorithm still performs reasonably well. In particular, the potential scale reduction factor (in the plots given in the right column) is relatively low and remains stable for the Republican dissimilarity index, suggesting that the impact of the starting values has mostly disappeared. Because African American voters are geographically concentrated, the algorithm has a harder time mixing for the African-American dissimilarity index. Nevertheless, the scale reduction factor still stabilizes at a reasonably low value, suggesting that the impact of the starting values is limited in this application.
4 Concluding Remarks

Over the last half century, a number of automated redistricting algorithms have been proposed in the methodological literature. Most of these algorithms have been designed to find an optimal redistricting plan given a certain set of criteria. However, many substantive researchers have been interested in characterizing the distribution of redistricting plans under various constraints. Unfortunately, few such simulation algorithms exist and even the ones that are commonly used by applied researchers have no theoretical justification.

In this paper, we propose a new automated redistricting simulator using Markov chain Monte Carlo. Unlike the existing standard algorithm, the proposed algorithms have a theoretical justification and approximate the target distribution well in a small-scale validation study. Even in more realistic settings where the computational challenge is greater, our initial analyses shows a promising performance of the proposed algorithms. Nevertheless, it is still unclear whether these algorithms scale to those states with an even greater number of districts than those considered here. In the future, we plan to investigate whether simulated and parallel tempering strategies can overcome the computational challenge posed by those large states.

Another promising line of research is to examine the factors that predict the redistricting outcome. For example, substantive researchers are interested in how the institutional features of redistricting process (e.g., bipartisan commission vs. state legislature) determines the redistricting process. Such an analysis requires inferences about the parameters that are underlying our generative model. In contrast, in this paper we restricted our attention to the question of how to simulate redistricting plans given these model parameters. Therefore, a different approach is required to address this and other methodological challenges.
Appendix: Proof of Theorem 1

Let $\Gamma(C^*, G_{v^*})$ denote all sets of connected components $C$ obtainable through “turning on” edges in $E_v$ such that $C^* \subset C$. Let $p(C \mid G_v)$ denote the probability that $C$ is obtained through Steps 1 and 2 of Algorithm 1. Let $p(C^* \mid C)$ denote the probability that, given $C$, its particular subset $C^*$ is selected at Step 3. Note that this probability does not depend on the partition $v_v$. Then, it follows that

$$\pi(v_{t-1} \rightarrow v_t^*) = \sum_{C' \in \Gamma(C^*, G_{v_{t-1}})} p(C^* \mid C')p(C' \mid G_{v_{t-1}}) \prod_{\ell=1}^{r} \frac{1}{|A(C^*_\ell, v_{t-1})|}$$  \hspace{1cm} (11)

$$\pi(v^*_t \rightarrow v_{t-1}) = \sum_{C' \in \Gamma(C^*, G_{v_t^*})} p(C^* \mid C')p(C' \mid G_{v_t^*}) \prod_{\ell=1}^{r} \frac{1}{|A(C^*_\ell, v_t^*)|}$$  \hspace{1cm} (12)

We now simplify equations (11) and (12) to identify common terms, which then cancel each other in equation (13). First, we show

$$|A(C^*_\ell, v_{t-1})| = |A(C^*_\ell, v_t^*)|$$  \hspace{1cm} (13)

for any connected component $C^*_\ell \in C^*$ where $l \in \{1, \ldots, r\}$.

Suppose that, without loss of generality, $C^*_\ell$ is adjacent to blocks $V_{1,t-1}, V_{2,t-1}, \ldots, V_{|A(C^*_\ell, v_{t-1})|+1,t-1} \in v_{t-1}$, and $C^*_\ell$ is contained in block $V_{|A(C^*_\ell, v_{t-1})|+1,t-1} \in v_{t-1}$. The check that $V^*_{k,t} \neq \emptyset$ in Step 4 of the algorithm ensures that $C^*_\ell \neq V_{|A(C^*_\ell, v_{t-1})|+1,t-1}$. Since $v_{t-1}$ is a connected set partition, there must exist $\{i|A(C^*_\ell, v_{t-1})|+1\} \in C^*_\ell$ and $\{j|A(C^*_\ell, v_{t-1})|+1\} \in V_{|A(C^*_\ell, v_{t-1})|+1,t-1} \setminus C^*_\ell$ that are adjacent in $G_{v_{t-1}}$. Moreover, there exist pairs of adjacent nodes $\{(i_1, j_1, \ldots, (i_1|A(C^*_\ell, v_{t-1})|+1), j_1)\}$ with $i_1 \in C^*_\ell, j_1 \in V_{k,t-1}$ where $1 \leq k \leq |A(C^*_\ell, v_{t-1})|$. Since $C^*$ is comprised of non-adjacent connected components, it follows that nodes $\{i_1, \ldots, j_1|A(C^*_\ell, v_{t-1})|\}, \{j_1|A(C^*_\ell, v_{t-1})|+1\}$ do not change block assignment when transitioning from $v_{t-1}$ to $v_t^*$, and thus, are contained in distinct blocks in $v_t^*$. Thus, the connected component $C^*_\ell$ is adjacent to all blocks corresponding to a node in $\{i_1, \ldots, j_1|A(C^*_\ell, v_{t-1})|\}, \{j_1|A(C^*_\ell, v_{t-1})|+1\}$ except for the block containing $C^*_\ell$: $|A(C^*_\ell, v_{t-1})|$ blocks in total. Hence, $|A(C^*_\ell, v_{t-1})| \geq |A(C^*_\ell, v_{t-1})|$. Moreover, for any block $V_{k,t-1} \notin A(C^*_\ell, v_{t-1})$ such that $C^*_\ell \not\subset V_{k,t-1}$, the corresponding block $V^*_{k,t}$ obtained by swapping connected components in $C^*$ will not be contained in $A(C^*_\ell, v_t^*)$; by definition, for any $\{i\} \in C^*_\ell, \{j\} \in V_{k,t-1} \setminus \{i,j\} \notin E$, and since connected components in $C^*$ are not adjacent, it follows that no edge connects a vertex in $V^*_{k,t}$ to a vertex in $C^*_\ell$. This proves equation (13).

Next, through a proof by contradiction, we show that

$$\Gamma(C^*, G_{v_{t-1}}) = \Gamma(C^*, G_{v_t^*}).$$  \hspace{1cm} (14)

By showing this, we also conclude that $v_{t-1}$ can be a candidate partition when starting from $v_t^*$, i.e., $\pi(v_{t-1} \rightarrow v_t^*) > 0$ implies $\pi(v_t^* \rightarrow v_{t-1}) > 0$. Suppose that there exists a set of connected components $C' \in \Gamma(C^*, G_{v_{t-1}})$ such that $C' \notin \Gamma(C^*, G_{v_t^*})$. This means that there exists $C' \in C'$ that can be formed by turning on edges in $E_{v_{t-1}}$, but not in $E_{v_t^*}$. Thus, there exists $\{i\}, \{j\} \in C' \in C'$ such that $(i,j) \in E_{v_{t-1}}$ and
follows that any set of edges connecting nodes in $V$ such that, when only those edges in a subset are turned on, the set of connected components $C$ is formed (Step 2). Note that $C$ can be formed if and only if the partition $\mathbf{v}$ satisfies $E_C \subset E_\mathbf{v}$, and $\Lambda(C, E_\mathbf{v})$ is identical for all such partitions. Specifically, $\Lambda(C, E_\mathbf{v}) = \Lambda(C, E_\mathbf{v}^*)$. To see this, observe that every set of edges $E_\mathbf{v}^* \in \Lambda(C, E_\mathbf{v})$ must connect nodes within each connected component in $C$, and must not include any edges joining a connected component to a node not included in the connected component. For any connected component $C_\ell \in C$, there must be a block $V_\ell \in \mathbf{v}$ such that $C_\ell \subset V_\ell$. Since $E_\mathbf{v}$ contains all edges joining two nodes in $V_\ell$, it follows that any set of connecting nodes in $C$ is contained in $E_\mathbf{v}$.

Third, we decompose $p(C \mid G_\mathbf{v})$. For a partition $\mathbf{v}$, let $\Lambda(C, E_\mathbf{v})$ denote all subsets of edges of $E_\mathbf{v}$ such that, when only those edges in a subset are turned on, the set of connected components $C$ is formed (Step 2). Note that $C$ can be formed if and only if the partition $\mathbf{v}$ satisfies $E_C \subset E_\mathbf{v}$, and $\Lambda(C, E_\mathbf{v})$ is identical for all such partitions. Specifically, $\Lambda(C, E_\mathbf{v}) = \Lambda(C, E_\mathbf{v}^*)$. To see this, observe that every set of edges $E_\mathbf{v}^* \in \Lambda(C, E_\mathbf{v})$ must connect nodes within each connected component in $C$, and must not include any edges joining a connected component to a node not included in the connected component. For any connected component $C_\ell \in C$, there must be a block $V_\ell \in \mathbf{v}$ such that $C_\ell \subset V_\ell$. Since $E_\mathbf{v}$ contains all edges joining two nodes in $V_\ell$, it follows that any set of connecting nodes in $C$ is contained in $E_\mathbf{v}$.

Given a set of “turned-on” edges $E_\mathbf{v}^* \in \Lambda(C, E_\mathbf{v})$, define $B(C^*, E_\mathbf{v}^*) = \{i, j\} \in E_\mathbf{v}^* : \{i\} \in C^*, \{j\} \notin C^*\}$ as the set of “turned-off” edges. Observe that, for $E_\mathbf{v}^* \in \Lambda(C, E_\mathbf{v})$, $E_\mathbf{v}^* \in \Lambda(C, E_{\mathbf{v}^*})$ with $E_\mathbf{v}^* = E_\mathbf{v}^* \backslash B(C^*, E_{\mathbf{v}^*})$ may be different from $E_\mathbf{v}^*$. That is, if the candidate partition $\mathbf{v}^*$ is obtained from $\mathbf{v}_{t-1}$ by assigning connected component $C' \in C$ from block $V_\ell$ to block $V_\ell'$, $E_{\mathbf{v}^*}$ may contain an edge that connects a node in $C'$ to an adjacent node in $V_\ell'$, whereas this edge cannot occur in $E_{\mathbf{v}^*}$.

Define

$$B(C^*, E_\mathbf{v}^*) = \{(i, j) \in E_\mathbf{v}^* : \{i\} \in C^*, \{j\} \notin C^*\}$$

as the set of edges in $E_\mathbf{v}^*$ that connect a block of nodes in $C^*$ to a vertex not in $C^*$, i.e., those edges that need to be “cut” to form blocks of vertices $C^*$. Since $C^* \subset C$, for partition $\mathbf{v}$, $B(C^*, E_\mathbf{v})$ is the same for every set of turned-on edges in $\Lambda(C, E_\mathbf{v})$, and is the same across all sets of connected components in $\Gamma(C^*, G_\mathbf{v})$. Then, we can write $p(C \mid G_\mathbf{v})$ as:

$$p(C \mid G_\mathbf{v}) = \prod_{e \in B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \sum_{E_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}})} \prod_{e \in E_{\mathbf{v}_{t-1}}} q_e \prod_{e \in \overline{E}_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e)$$

(16)

where we allow the edge cut probability to differ across edges.

Finally, we show that, for any $E_{\mathbf{v}_{t-1}}^* \in \Lambda(C, E_{\mathbf{v}_{t-1}})$, $E_{\mathbf{v}_{t-1}}^* \in \Lambda(C, E_{\mathbf{v}_{t-1}})$ with $E_{\mathbf{v}_{t-1}}^* = E_{\mathbf{v}_{t-1}}^*$:

$$E_{\mathbf{v}_{t-1}}^* \setminus B(C^*, E_{\mathbf{v}_{t-1}}) = E_{\mathbf{v}_{t-1}}^* \setminus B(C^*, E_{\mathbf{v}_{t-1}})$$

(17)

Consider any edge $e \in E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})$. This edge can either join two nodes within a single connected component or joins two nodes in two distinct connected components. In the former case, both nodes are contained in a single block of $\mathbf{v}_{t-1}$,
and since connected components are reassigned to form the candidate partition \( \mathbf{v}^* \), it follows that both nodes are contained in a single block \( V^* \in \mathbf{v}^* \). Hence, \( e \in E_{\mathbf{v}^*} \), and since does not join a node in connected component in \( C^* \) to a node in a connected component that is not in \( C^* \), it follows that \( e \in E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \). In the latter case, observe that, since \( e \in E_{\mathbf{v}^*_t} \), both connected components must be contained within the same block of \( \mathbf{v}_{t-1} \). Since they do not belong to \( C^* \), neither component is reassigned to a block, and hence, are contained within the same block \( V_{kt} \in \mathbf{v}^*_t \). Thus, \( e \in E_{\mathbf{v}^*_t} \), and since does not join a node in connected component in \( C^* \) to a node in a connected component that is not in \( C^* \), it follows that \( e \in E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \). In both cases, \( e \in E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \). Thus, \( E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \subset E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \). By the same argument, \( E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \subset E_{\mathbf{v}^*_t} \setminus B(C^*, E_{\mathbf{v}^*_t}) \), and thus, we have shown equation (17). By this observation, we can now write,

\[
p(C \mid G_{\mathbf{v}^*_t}) = \prod_{e \in B(C^*, E_{\mathbf{v}^*_t})} (1 - q_e) \sum_{E_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}}) \in E_{\mathbf{v}_{t-1}}} q_e \prod_{e \in E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e).
\]

Using equation (16) and the fact that the set of edges \( B(C^*, \mathbf{v}_{t-1}) \) is identical across all sets of connected components \( C_t \in C^* \), we can write as:

\[
\pi(\mathbf{v}_{t-1} \rightarrow \mathbf{v}^*_t) = \prod_{e \in B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \sum_{C \in \Gamma(C^*, \mathbf{v}_{t-1})} \left( \prod_{E_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}}) \in E_{\mathbf{v}_{t-1}}} q_e \prod_{e \in E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \right) \times p(C^* \mid C) \prod_{\ell=1}^r \frac{1}{|\Lambda(C^*_\ell, \mathbf{v}_{t-1})|}.
\]  

Similarly, we find that:

\[
\pi(\mathbf{v}^*_t \rightarrow \mathbf{v}_{t-1}) = \prod_{e \in B(C^*, E_{\mathbf{v}^*_t})} (1 - q_e) \sum_{C \in \Gamma(C^*, \mathbf{v}_{t-1})} \left( \prod_{E_{\mathbf{v}_{t-1}} \in \Lambda(C, E_{\mathbf{v}_{t-1}}) \in E_{\mathbf{v}_{t-1}}} q_e \prod_{e \in E_{\mathbf{v}_{t-1}} \setminus B(C^*, E_{\mathbf{v}_{t-1}})} (1 - q_e) \right) \times p(C^* \mid C) \prod_{\ell=1}^r \frac{1}{|\Lambda(C^*_\ell, \mathbf{v}_{t-1})|}.
\]

Thus, many terms cancel out and we obtain the following expression for the acceptance probability:

\[
\alpha(\mathbf{v} \rightarrow \mathbf{v}^*) = \min \left( 1, \frac{\prod_{e \in B(C^*, \mathbf{v}^*_t)} (1 - q_e)}{\prod_{e \in B(C^*, \mathbf{v}_{t-1})} (1 - q_e)} \right).
\]

In the special case that edges are turned on with equal probability, i.e., \( q = q_e \) for all \( e \), this ratio can be computed by counting the number of edges connecting nodes in blocks of \( C^* \) to nodes outside of those blocks:

\[
\alpha(\mathbf{v} \rightarrow \mathbf{v}^*) = \min \left( 1, \frac{(1 - q)^{|B(C^*, \mathbf{v}^*_t)|} - |B(C^*, \mathbf{v}_{t-1})|}}{(1 - q)^{|B(C^*, \mathbf{v}_{t-1})|} - |B(C^*, \mathbf{v}_{t-1})|} \right).
\]

\( \square \)
References


